THE MIMOSA GENERIC MODELLING AND SIMULATION PLATFORM: THE CASE OF MULTI-AGENT SYSTEMS

Jean-Pierre Müller*
CIRAD-TERA-REV-GREEN
73, av. Jean-François Breton, F-34398 Montpellier cedex 5, France
E-mail: jean-pierre.muller@cirad.fr

KEYWORD
simulation platform, multi-formalism, multi-agent system

ABSTRACT

The aim of this paper is to introduce Mimosa: a generic platform for modelling and simulation and to show its applicability to the specification of multi-agent simulations based on the AGR (Agent-Group-Role) model. The Mimosa platform allows to describe the appropriate formalisms for the targeted models and then the models themselves. Moreover, it is able to articulate several models for modelling complex systems. This capability is illustrated by showing how the notions of environment, agents, groups and roles are specified, instantiated into concrete models and then run through the specification of the notion of scheduler. Finally, an running example is given.

INTRODUCTION

The aim of the Mimosa(mimosa, 2003) project is to specify the tools allowing a modeler to describe his models in his appropriate formalism as well as the simulations of these models. Most of the existing modelling and simulation platforms define the formalism or the concepts in which the model is expressible (Stella(Tilideske, 1998), Devs(Zeigler et al., 1999), Cormas(cormas, 2003), Starlogo(starlogo, 2002), etc.) or partly give the liberty to define the desired concepts by giving direct access to the underlying programming language as a kind of universal formalism (Swarm(swarm, 2003), MadKit(madkit, 2003), RePast(repast, 2003)). Our goal is to allow the flexibility to define the most appropriate formalisms for model description without the need to access the underlying programming language. Consequently, a generic modelling tool should allow to both define the discourse (i.e. the model) and the way to express it (the formalism). Modelling a complex system as, for example, an eco-sociosystem requires the multiplicity of points of view (ecological, agronomical, sociological, economical, etc.) which are each a specific discourse with its own mean of expression and which have to be articulated together. We insist on articulation rather than integration which would assume or impose a kind of universal and ultimate formalism in which everything could be expressed.

In order to fullfill these aims, Mimosa is composed of two layers. The basic layer is a toolbox providing the elements able to build any formalism (cellular automata, compartment models, state-charts or multi-agent system for dynamical models, conceptual graphs, spatial and temporal structures for more structured descriptions). The second layer is an open-ended set of plugins specializing the toolbox for specific purposes and providing the modeler with the means to express his models. As an example, multi-agent systems are composed of:

- an environment which can be as simple as a communication graph up to complex environments populated with objects and provided

*This reasearch was partially made under the Swiss FNRS grant no 2153-63958.00

with its own dynamics;

- and the agents which are locally interacting entities between themselves and with an environment. In the AGR (Agent-Group-Role(Ferber and Gutknecht, 1998)) model, the social interactions are modeled through groups in which the agents are playing specific roles.

For simulation, we have additionally to introduce explicitly the notion of time and to describe the way time flows in the simulation. Therefore, we have to explain how the basic layer of Mimosa can be used to specify the means to describe environments, agents, groups, roles and time.

The next section shall introduce the state of the art in simulation and modelling platforms. In the next section, we shall introduce the basic layer of Mimosa. This basic layer shall be instantiated specifically for AGR multi-agent systems. An example shall be given before concluding with a summary and some perspectives.

STATE OF THE ART

It is not possible to describe all the modelling and simulation platforms because but we will concentrate on the main available tools for multi-agent simulation used by an important user community. We shall divide the existing platforms in two categories:

1. The generic platforms making almost no assumption on the kind of model the modeler is designing by giving direct access to the underlying programming language;

2. The dedicated platforms providing a formalism or set of concepts in which the models have to be expressed.

In the first category, the most well known multi-agent platform is Swarm, which has been initially realized by Chris Langton at the Santa Fe Institute and now by an important community of developpers (Daniels, 1999; swarm, 2003). Delivered with a set of class libraries in Objective C (and now in Java), Swarm is very efficient in terms of performance and existing tools but requires a long learning curve due to its low level API at the programming language level. Repast(Collier, 2003), following Ascape, is as swarm-like simulation environment. A model is described by Java classes (possibly generated separately by a simulation building tool) and loaded into the system for execution. In the same vein, one can cite the Mason platform which intends to be extendable through plugins to interface with external systems or languages (Luke et al., 2003).

In the second category, we have StarLogo(starlogo, 2002; Resnick, 1994; Reisnick, 1996) which is designed with a dedicated language for time-step simulations and has some successors like NetLogo and Breve(breve, 2003; Klein, 2002) which extends the possibilities for object-oriented description (using the steve language), continuous 3D simulation and some discrete event capabilities. Cormas(Bousquet et al., 1988), developped at CIRAD-Montpellier, provides the building blocks for describing cellular automata made of patches, agents and their interactions (communicating and/or situated agents). However, the behaviours have to be programmed in Smalltalk. Even more specialized modelling and simulation systems exist like Stella(Tilideske, 1998) for compartment models and its integration in SME(Voinov et al., 1992), or Devs(Zeigler et al., 1999) for multi-modeling.

MadKit(madkit, 2003; Ferber and Gutknecht, 1998; Gutknecht and Ferber, 2000) is a generic multi-agent platform written in Java which is used both for multi-agent research in general and for simulation through a dedicated synchronous engine which is similar to starlogo. This software falls in between the two categories by its programming language level of description and the definition of dedicated mechanisms for simulation.

If the problem of multi-modeling is widely recognized, most of the existing solutions are proposed in dynamical modelling community like with SME(Voinov et al., 1992) or Devs(Zeigler et al., 1999), even if the later proposes encapsulation of heterogeneous models(Ramat and Preux, 2003). If programming level platforms allows it in principle (Swarm, MadKit), there is no general mechanism for doing so. As soon as the tools are designed at modeler language level (StarLogo, Stella, Devs), they are strictly restricted to one kind of formalism. Therefore, there is place for a generic simulation platform which is:

at the modeler level: providing suitable for-

malisms to let the modeler express his models as he wants;

extensible: as for Swarm or Mason but with the possibility to add further modeler level formalisms;

interoperable: in the sense that the various models can work together in the same simulation, which is not the case for any available platform as far as we know.

THE BASIC LAYER

The basic layer of MIMOSA is a toolbox to describe the structure of the formalism and its dynamics. We shall present these two aspects successively.

The structure

Basically, any formalism is made of parts and wholes. For example, a space is made of places, a state-chart is made of states, a cellular automata is made of cells, etc.. Consequently, we have introduced the notion of *component* for describing the parts seen as indecomposable (atomic) and the notion of *compound* for describing the wholes seen as sets of components. The structure is not recursive as compounds cannot be themselves components. Additionally the compounds introduce the way to name their components. For example, a space names its components (places) either by names (Lisbon, France, etc.) or by coordinates.

In addition, we introduce three notions of *relations* for various purposes:

intra-compound relations: are relations between the components of a compound. They can be used to describe the transitions between the states, the relationships between places (adjacency, containment, etc.) or the neighbourhood relation among the cells of a cellular automaton;

inter-compound relations: are relations between two compounds (i.e. relations between two sets). For example, they can be used to describe the mapping of one space into another;

component-compound relations: are relations between a component and a compound. They are used to describe that an atomic component from one point of view can be seen as compounded from another point of view. They provide the possibility of introducing explicity the recursivity.

The dynamics

For introducing the dynamics, any object of the system (components, compounds and relations) is provided with an internal *state* which is encapsulated (in the sense that it is not accessible from the outside of the object) and can evolve over time. Additionally, any object of the system can be measured by using *measure tools* and can change state in reaction to *events*. A measure tool can be parameterized to specify when, where and how (scale, etc.) the measure has to be taken. It gives access to information on the object state. The events can also be time stamped and they systematically are when used for running a simulation. An event usually results in a state change and the production of new events. The choice of using events calls for discrete event simulation mechanisms. However, fixed time step simulation can be made by using clocks explicitly as generators of stamped events. Moreover, the use of state and events as continuous functions of time are envisioned for articulating both continuous and discrete time. Currently, the management of the events and the measure requests is delegated to specific Java classes. In a future version, it will be possible to specify the actual behaviour with scripting or rule-based languages, or, for components, by relating them to a compound implementing automata or compartment models.

summary

The basic layer is build on the notions of component, compound, relation, state, measure tool and event which appear to be very minimal. Moreover, we introduce the distinction between the types and the instances. Therefore, the modeler must define his ontology by defining the object types build with the appropriate formalism and then create his models from these types.

IMPLEMENTING MULTI-AGENT SYSTEMS

As described in the introduction, we shall successively describe the environment, the agents, the groups, the roles and a simplified account of time.

The environment

In multi-agent systems, the environment is generally considered as a network of places. Only a few systems introduce continuous spaces and we will not consider this case in this paper. We have designed a plugin with various environment cases:

- the space made of named places in which case the components are the places named by symbols and the compounds are the possible spaces. An intra-compound relation is defined in order to describe the neighbourhood relation;
- the one-dimensional grid where the places are named by a single coordinate. A neighbourhood relation is defined as an intra-compound relation with an optional toroidal feature allowing to describe circular spaces;
- the two-dimensional grid where the places are named by two coordinates. A neighbourhood relation is also defined as an intra-compound relation with an optional toroidal feature allowing to describe toroidal spaces.

Regarding the dynamics, the spaces are until now considered as static (no place is added or removed) but each place can be provided with a state and a state transition function called on the incoming of adequate events (now the events for initializing and for synchronous and asynchronous updating are provided). We obtain this way a slighltly generalized cellular automaton on which the agents can be situated as in Cormas.

The agents

An agent can be seen as a component of a population of agents (the compound). Therefore we introduce both the notion of population and agent. No relations are defined on the population although an aquaintance relation could be defined for message passing. However, we have chosen another strategy for communication based on the AGR (Agent-Group-Role(Ferber and Gutknecht, 1998)) principles as will be described in the next section.

Regarding the dynamics, the population can grow and shrink in response to related events. Additionally, each agent is itself provided with a state which for the time being also reacts to initialize, synchronous and asynchronous updates for uniformity with the cellular automata defined in the previous section as possible environments.

In order for an agent to be properly situated in an environment, we define an inter-compound relation between any population of agents and any space. This relation, called *position*, defines which agent is on which place and therefore, maps agents into places. The position relation reacts to the *move* events (among others) and change the place the agent is situated on if the the new place is not already occupied. This behaviour naturally implements the influence/reaction model where agents only proposes changes (the influences) and the environment actually perform the resulting (and possibly combined) change. This is achieved at the level of the relation rather than of the environment as suggested in (Ferber and Müller, 1996).

This presentation calls for some comments. First, an agent can be situated in several environments at the same time. Each environment and relation to the environment defines what it is able to perceive (through the measuring tools) and what it is able to do (through the available events). If these environments happen to be various points of view on the same one, they can be implemented either as filters on a shared representation or on consistency constraints between the points of view. In either case, the inter-compound relations have to be used. Second, the influence/reaction model is no longer centralized in teh dynamics of a single centralized environment but distributed among several environments as long as the agent intervenes in each environment proper dynamics and several relations as long as the agent acts on its relationships with the environments.

The groups and roles

In the AGR model, an agent can only communicate within a group by playing a role and can play several roles in several groups at the same time

(said otherwise, two groups can only communicate through a shared agent: the notion of representative). Therefore when an agent wants to communicate, it must create a new group or enter an existing group. The formulation in Mimosa is straightforward. The components are the roles and the compounds are the groups. Each role (as each group) has a name like in MadKit. A *participation* relation maps the set of agents to the set of roles specifying which agent plays which role. An instance of this relation is created for each group and allows the agent to send messages (a particular kind of event). The role receives the message and send it to the related role which posts it into its related agent mailbox. In the current implementation, the roles only specify the destination of the outgoing messages. Another possibility is to embed into the roles the full interaction protocols, allowing to describe them separatly from the agents and therefore making them reusable. This approach has been proposed in (Hilaire et al., 2000) and fully implemented in (Amiguet et al., 2002; Amiguet, 2003). We intend to port this implementation into Mimosa.

The scheduler

As described in the previous sections, the places, the agents and even the roles in a future version, can receive events for initializing and for performing synchronous or asynchronous updates. They could also receive arbitrary time-stamped events in which case each compound would have to play the role of a discrete event simulation scheduler (reproducing in a simple way the notion of schedule found in Swarm). This later possibility has not been implemented yet. The actual implementation allows to have clock components generating tick events with a fixed duration. A scheduler is implemented has a compound with one clock component and a number of clock-controlled components. When a clock component receives an event for advancing a number of steps, it broadcasts each tick event to the set of clock-controlled components. The clock-controlled components can be linked to any other component or compound to which it will request for either a synchronous or an asynchronous update. This implementation provides the possibility to cascade schedulers as in Swarm.

summary

In this section, we have described in detail how we have implemented the plugins for describing the environments, the populations of agents and the groups and roles by specializing the generic notions of the basic layer of Mimosa. Given the possibility to model these notions, we shall use them into a concrete example.

AN EXAMPLE

To illustrate the use of the various modelling formalisms we have introduced in the previous section, we shall describe the fire man application:

- the environment is a cellular automata computing the propagation of fire in a forest. The state of a cell is either empty, with trees, in fire or wet. A cell initially has a given probability to be occupied by trees. At each time step, a cell with trees has a given probability to burn. If there is fire in the direct neighbourhood, a cell with trees also burns otherwise it stays unchanged;

- a population of fireman agents has been defined. Each fireman walks around randomly until he finds a fire in its direct neighbourhood in which case it pours water on the given cell. Moving events are sent to the position relation, pouring water events are sent to the places. It is up to the relation to actually perform the move if possible and up to the place to decide whether to stop burning or not (influence/reaction);

- a fireman near a fire creates a group in which it enters the role of coordinator to request help. The unoccupied firemen look for existing groups and enter them as coordinated to receive the coordinator commands. They will move to the nearest coordinator;

- a scheduler is defined with one clock and two clock-controlled components in order to run synchronously both submodels: the fireman team and the cellular automaton. The clock events are time stamped in order to be independent of the activation order between the multi-agent system and the environment.

The figure 1 shows the output of the running simulation with the MIMOSA model: the building window in the background and the three windows for the cellular automaton, the fire team and the scheduler. At this stage, the agents are not yet visible on the cellular automaton view and better visualization tools still have to be designed.

This example has been extended to the implementation of aggregates. An aggregate is a place which is itself a set of adjacent places sharing a common property. In our case, we have created a space made of aggregates which is related to the cellular automaton by an appropriate inter-compound relation. This relation updates the set of aggregates and their composition according to the changes to the state of the cells such that each aggregate is formed of adjacent burning cells. The aggregate space is another environment in which the agent can ask for the aggregate size and only requests help when the fire becomes too big for him alone. This notion of aggregation has already been implemented in Cormas and found an easy definition in Mimosa.

CONCLUSION

We have presented the MIMOSA framework which is a generic modelling and simulation platform providing the means to define the formalisms in which models must be described, the modeler ontologies and then the models themselves. The formalisms are build out of a very minimal toolbox made of components, compounds, relations, states, measuring tools and events. The usefullness of this toolbox to design non trivial models has been demonstrated by instantiating these concepts for designing a very general multi-agent system: at least as expressive as MadKit, Cormas or Swarm. Of course this development is very new. The underlying mechanisms must be completed with a generalized discrete event management and the capability to mix continuous and discrete time. The set of plugins is open-ended but some basic one must be implemented like the time representations and the graphical interfaces.

We would like to thank all the participants of the Mimosa working group(mimosa, 2003) and of the Green team(cormas, 2003) for very useful discussions on the underlying concepts of the current platform.

REFERENCES

Amiguet, M. (2003). *MOCA: un modèle componentielle dynamique pour les systèmes multi-agents organisationels*. PhD thesis, Université de Neuchâtel.

Amiguet, M., Müller, J.-P., Baez-Barranco, J., and Nagy, A. (2002). The moca platform: Simulating the dynamics of social networks. In *MABS'02*. Springer Verlag.

Bousquet, F., Bakam, I., Proton, H., and Page, C. L. (1988). *Cormas: Common-Pool Resources and Multi-Agent Systems*, pages 826–837. Number 1416 in LNAI. Springer Verlag.

breve (2003). http://www.spiderland.org/breve/.

Collier, N. (2003). Repast: An extensible framework for agent simulation. Technical report, University of Chicago.

cormas (2003). http://cormas.cirad.fr/.

Daniels, M. (1999). Integrating simulation technologies with swarm. In *proceedings of "Agent Simulation: Applications, Models, and Tools"*.

Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings ICMAS '98*.

Ferber, J. and Müller, J.-P. (1996). Influences and reaction: a model of situated multiagent systems. In *ICMAS'96*.

Gutknecht, O. and Ferber, J. (2000). Madkit: a generic multi-agent platform. In *Autonomous Agents (AGENTS 2000)*, pages 78–79. ACM Press.

Hilaire, V., Koukam, A., Gruer, P., and Müller, J.-P. (2000). Formal specification and prototyping of multi-agent systems. In *ESAW'2000*.

Klein, J. (2002). Breve: a 3d simulation environment for the simulation of decentralized systems and artificial life. In *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems*. MIT Press.

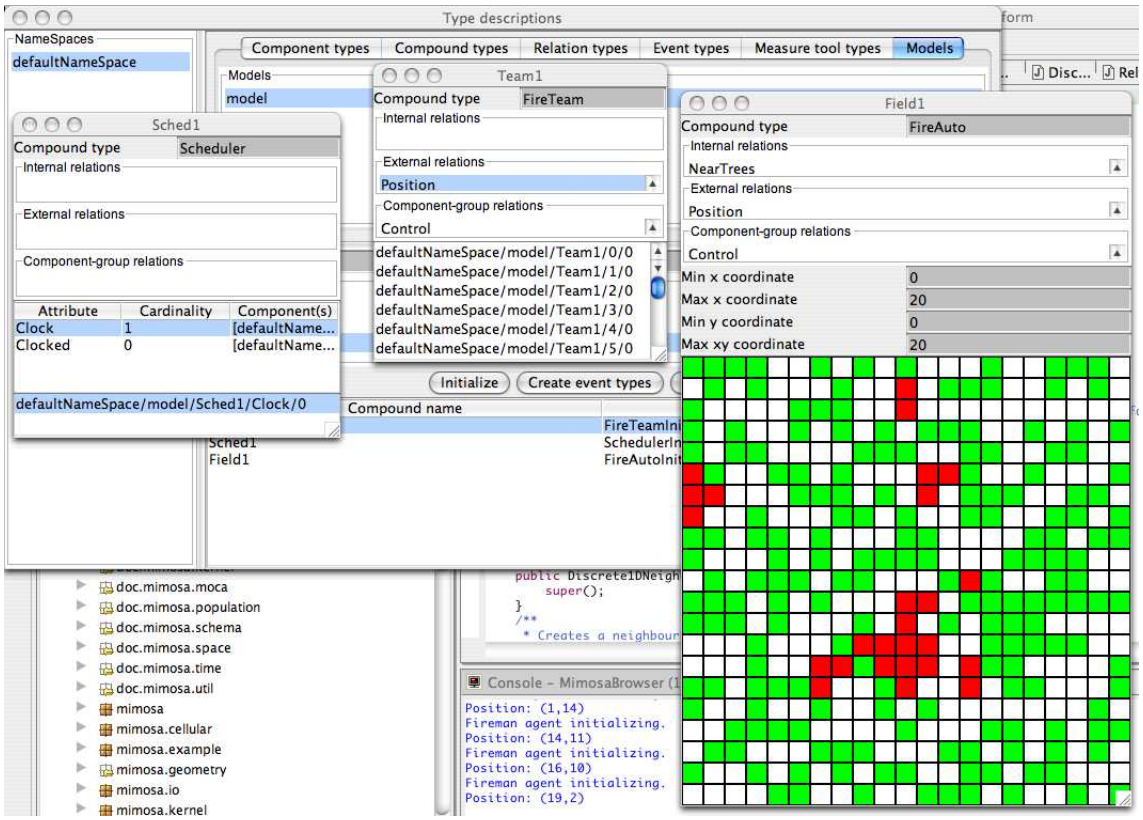

Figure 1: The fireman model

Luke, S., Balan, G. C., Panait, L., Cioffi-Revilla, C., and Paus, S. (2003). Mason: A java multi-agent simulation library. In *Proceedings of the Agent 2003 Conference*.

madkit (2003). http://www.madkit.org/.

mimosa (2003). http://www-lil.univ-littoral.fr/mimosa/.

Ramat, E. and Preux, P. (2003). Virtual laboratory environment (vle): a software environment oriented agent and object for modeling ans simulation of complex systems. *Simulation Modelling Practice and Theory*, 11:45–55.

Reisnick, M. (1996). Starlogo: An environment for decentralized modeling and decentralized thinking. In *Proceedings of CHI'96*.

repast (2003). http://repast.sourceforge.net/.

Resnick, M. (1994). *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press.

starlogo (2002). http://education.mit.edu/starlogo/.

swarm (2003). http://www.swarm.org/.

Tilideske, B. (1998). Modeling and simulation using stella. Technical report, Charleston Southern University.

Voinov, A., Fitz, C., Maxwell, T., Boumans, R., and Costanza, R. (1992). Modular ecosystem modeling. In *Porceedings of iEMSs02*.

Zeigler, B., Kim, D., and Praehofer, H. (1999). *Theory of modeling and simulation: 2nd edition*. John Wiley.